II. AMENDMENTS TO THE SPECIFICATION

Please replace the specification of the present application with the following amendments:

In ¶[0001], please amendment as follow: -

[0001] The present invention generally relates to a system, method and program product for dynamically adding resources to, or updating resources in, a web application. Specifically, the present invention provides a way to maintain session information when dynamically adding [[Java]]-JAVA™ resources to, or updating [[Java]] JAVA™ resources in, a web application ("JAVA" is a trademark of Sun Microsystems, Inc., Santa Clara, CA.).

In ¶[0003], please amendment as follow: -

[0003] In view of the foregoing, there exists a need for a system, method and program product for dynamically adding [[Java]] JAVA™ resources (e.g. [[Java]] JAVA™ ResourceBundles, class files, properties, images, etc.) to, or updating [[Java]] JAVA™ resources in, a web application without requiring the web application or a corresponding web application server to be restarted and without losing session state.

In ¶[0004], please amendment as follow: -

[0004]In general, the present invention provides a system, method and program product for dynamically adding [[Java]] JAVA™ resources to, or updating [[Java]] JAVA™ resources in, a web application without requiring the web application or a corresponding web application server to be restarted and without losing session state. Specifically, under the present invention, to preserve session information and prevent restarting of the end user web application, [[Java]] JAVA™ resources (e.g., [[Java]] JAVA™ ResourceBundles) are not obtained directly (e.g., via

the [[Java]] JAVA™ ResourceBundle.getBundle call), which would retrieve the [[Java]] JAVA™ resources from the class loader of the web application. Rather, [[Java]] JAVA™ resources are obtained indirectly from a separate web application (i.e., a Resource Lookup Web Application (RLWA)), whose only duty is to serve [[Java]] JAVA™ resources.

In ¶[0006], please amendment as follow: -

[0006]A second aspect of the present invention provides a method, comprising: dynamically adding a [[Java]] JAVA™ resource to at least one web application in a web application server without having to restart the web application and without losing session information in the web application.

In ¶[0009], please amendment as follow: -

[0009]Therefore, the present invention provides a system, method and program product for dynamically adding [[Java]] JAVA™ resources to, or updating [[Java]] JAVA™ resources in, a web application without requiring the web application or a corresponding application server to be restarted and without losing session state.

In ¶[0015], please amendment as follow: -

[0015] As indicated above, the present invention provides a system, method and program product for dynamically adding [[Java]] JAVA™ resources to, or updating [[Java]] JAVA™ resources in, a web application without requiring the web application or a corresponding application server to be restarted and without losing session state. Specifically, under the present invention, to preserve session information and prevent restarting of the end user web application, [[Java]] JAVA™ resources (e.g., [[Java]] JAVA™ ResourceBuldles) are not obtained directly (e.g., via the [[Java]] JAVA™ ResourceBundle.getBundle call), which would retrieve the

[[Java]] JAVA™ resources from the class loader of the web application. Rather, [[Java]] JAVA™ resources are obtained indirectly from a separate web application (i.e., a Resource Lookup Web Application (RLWA)), whose only duty is to serve [[Java]] JAVA™ resources.

In ¶[0017], please amendment as follow: -

[0017] RLWA 14 includes [[Java]] JAVA™ resources 16, which may include ResourceBundles, class files, etc. On startup, RLWA 14 places a reference to each [[Java]] JAVA™ resource 16 (or a way to get to each [[Java]] JAVA™ resource 16) in its servlet context with the Servlet API call "getServlet().setAttribute(resourceName, resource)," where "resourceName" is the name of the [[Java]] JAVA™ resource and "resource" is the object to be referenced dynamically. The Servlet API call "getServlet().setAttribute(resourceName, resource)," which is a well known call in a [[Java]] JAVA™ based web application server environment, and places a tag or similar identifier corresponding to each available [[Java]] JAVA™ resource 16 in a table that can be accessed by each web application 12A-C. In this manner, i.e., via its servlet context, the RLWA 14 "advertises" the [[Java]] JAVA™ resources 16 it has available to each web application 12A-C within web application server 10.

In ¶[0018], please amendment as follow: -

[0018] Each web application 12A-C can obtain a [[Java]] JAVA™ resource 16 from the RLWA 14 servlet context via the Servlet API call "getServletContext().getContext(url)," where "url" is the path to RLWA 14's servlet as defined in its web.xml file. This allows each web application 12A-C to see which [[Java]] JAVA™ resources 16 are available from RLWA 14. From that context, each web application 12A-C can call "getAttribute(resourceName)," where "resourceName" is the name of the desired [[Java]] JAVA™ resource. Referring to FIG. 1,

arrows 18 represent [[Java]] JAVA™ resource 16 requests from web applications 12A-C to RLWA 14, while arrows 20 represent RLWA 14 responding with the requested [[Java]] JAVA™ resources 16. Again, the calls "getServletContext().getContext(url)" and "getAttribute(resourceName)," are commonly used to obtain [[Java]] JAVA™ resources in a [[Java]] JAVA™ based web application server environment.

In ¶[0019], please amendment as follow: -

[0019] A new or updated [[Java]] JAVA™ resource 16' (e.g., a ResourceBundle, class file, etc.), supplied by an external source, can be added in a known manner to RLWA 14 and loaded dynamically. While this will invalidate any session information on RLWA 14, it will not invalidate the session information for web applications 12A-C. Specifically, after the new or updated [[Java]] JAVA™ resource 16' has been added to RLWA 14, [[Java]] JAVA™ resource 16' is available to all of the web applications 12A-C and can be requested by, and provided to, a requesting web application 12A-C from RLWA 14 in the manner detailed above. The requesting web application 12A-C subsequently receives and uses the new or updated ~~[[Java]]~~ JAVA™ resource 16' without losing session information and without having to be restarted. In this way, the new or updated [[Java]] JAVA™ resource 16' can be used seamlessly and appear in the user interface of the requesting web application 12A-C without end users experiencing any downtime or loss of information. In other words, under the present invention, to preserve session information and prevent restarting of the web application 12A-C, new/updated [[Java]] JAVA™ resources 16' are not obtained directly (i.e., via the class loader of the web application), but rather indirectly from a separate web application (i.e., RLWA 14), whose only duty is to serve [[Java]] JAVA™ resources.

In ¶[0020], please amendment as follow: -

[0020] Typically, prior to the present invention, each web application in a web application server was required to load and install [[Java]] JAVA™ resources individually. Under the present invention, however, RLWA 14 manages [[Java]] JAVA™ resources 16, 16' for all of the web applications 12A-C, such that each [[Java]] JAVA™ resource 16, 16' is installed once at a single location and is commonly accessible to any web application 12A-C running in the web application server 10. Therefore, [[Java]] JAVA™ resources 16, 16' can be installed once in a single location in web application server 10, thereby reducing memory overhead, and then shared by any web application 12A-C residing in the same web application server 10.

In ¶[0024], please amendment as follow: -

[0024]Storage unit 42 can be any system capable of providing storage for information, such as [[Java]] JAVA™ resources 16, 16', under the present invention. As such, storage unit 42 may reside at a single physical location, comprising one or more types of data storage, or be distributed across a plurality of physical systems in various forms. In another embodiment, storage unit 42 may be distributed across, for example, a local area network (LAN), wide area network (WAN) or a storage area network (SAN) (not shown).

In ¶[0025], please amendment as follow: -

[0025] Shown in memory 34 of web application server 10 is RLWA 14, which includes resource receiving system 50 for determining when new or updated [[Java]] JAVA™ resources 16' are available and for obtaining the new or updated [[Java]] JAVA™ resources 16'. In particular, when a new/updated resource is available, web application server 10 restarts the RLWA 14, and its initialization method is invoked. Resource receiving system 50 of RLWA 14 then checks for and

obtains the new/updated resource during the processing of RLWA 14's initialization method. RLWA 14 further includes a resource install system 52 for dynamically installing new/updated [[Java]] JAVA™ resources 16' into RLWA 14, a [[Java]] JAVA™ resource advertising system 54 for advertising the availability of new/updated [[Java]] JAVA™ resources 16' to the web applications running in web application server 10 (e.g., web applications 12A-C, FIG. 1) via the Servlet API call "getServlet().setAttribute(resourceName, resource)" or other known manner. Although the dynamic installation/loading of the new/updated [[Java]] JAVA™ resources 16' into RLWA 14 will invalidate any session information on RLWA 14, it will not invalidate the session information for web applications 12A-C. Once the new/updated [[Java]] JAVA™ resources 16' have been added to RLWA 14, new/updated [[Java]] JAVA™ resources 16' are now available to all of the web applications 12A-C and can be requested from web application server 10 via resource request system 56. For example, the new/updated [[Java]] JAVA™ resources 16' can be requested by, and provided to, a requesting web application 12A-C from RLWA 14 using the above-described "getServletContext().getContext(url)" and "getAttribute(resourceName)" calls, or using other known techniques. A requesting web application 12A-C subsequently receives and uses the requested new or updated [[Java]] JAVA™ resource 16' without losing session information and without having to be restarted. In this way, the new or updated [[Java]] JAVA™ resource 16' can be used seamlessly and appear in the user interface of the requesting web application 12A-C without end users experiencing any downtime or loss of information. Indeed, an end user of the requesting web application 12A-C may never be aware that the above-described process has actually taken place.

In ¶[0026], please amendment as follow: -

[0026]Referring now to Fig. 3, a method flow diagram 100 according to the present invention is shown. As depicted, in step S1, RLWA 14 obtains a new or updated [[Java]] JAVA™ resource 16' when it become available. In step S2, RLWA 14 installs the new/updated [[Java]] JAVA™ resource 16'. Next, in step S3, RLWA 14 "advertises" the availability of the new/updated [[Java]] JAVA™ resource 16' to the web applications (e.g., web applications 12A-C, FIG. 1). This can be achieved by the RLWA 14, for example, using the Servlet API call "getServlet().setAttribute(resourceName, resource)." In step S4, one or more web applications request the updated [[Java]] JAVA™ resource 16' via the "getServletContext().getContext(url)" and "getAttribute(resourceName)" calls. Thereafter, in step S4, the requesting web application receives, dynamically adds, and uses the requested new/updated [[Java]] JAVA™ resource 16'.